Unele versiuni noi ale limbajului Lisp (Scheme şi Common-Lisp spre exemplu) au adoptat **legarea statică a domeniului**.

Exemplu (Common-Lisp):

```
>(defun f1(x)
   (f))

>(defun f( )
   x)

>( f1 5 )

*** - EVAL: variable X has no value
```

Explicația semnalării erorii este aceea că valoarea variabilei `x` din funcția `f` este căutată static în mediul funcției `f` şi apoi în mediul înconjurător textual care este mediul global. Deoarece ea nu este definită acolo, ci doar în mediul local funcției `f1`, se semnalează eroare.

Din motive de compatibilitate cu dialectele Lisp mai vechi, în Common-Lisp există şi posibilitatea legării unui domeniu dinamic la o variabilă prin solicitarea explicită a programatorului. În aceste cazuri, legarea se caută, nu în mediul înconjurător textual, ci în mediile parcurse anterior, oriunde ar fi ele în program.

Exemple :

1) Declararea unei variabile locale ca `special`

```
>(defun f1(x)
   (declare (special x))
   (f))
>(defun f( )
   x)
>(f1 5)
5
```


2) Declararea globală a unei variabile, cu `defvar`

```
>(defar x)
>(defun f1(x)
    (f))
>(defun f(x)
    x)
>(f1 5)
5
```

Durata de existență a variabilei. Alocarea memoriei

Durata de existență se referă la intervalul de timp cât o anumită zonă de memorie este asociată variabilei.

Asocierea zonei de memorie corespunzătoare unei variabile se realizează prin aşa numita *alocare*.

Alocarea poate avea loc:

- **static**: înainte de execuție; o anumită zonă, stabilită la compilare, rămâne asociată variabilei pe tot parcursul execuției programului;
- **dinamic**: alocarea are loc în timpul execuției programului; zona alocată poate fi eliberată ulterior.

  Alocarea dinamică, la rândul ei, se face:
  - **automat**: fără solicitare din partea programatorului;
  - **la cerere**: ca urmare a solicitării explicite din partea programatorului, printr-o instrucțiune corespunzătoare (`new`, `malloc`).